Universidade do Porto

Faculdade de Engenharia

FEUP

Whatsapl

AEDA 2015/2016

António Melo

up201403053@fe.up.pt

Bruno Santos

up201402962@fe.up.pt

23/12/2015

Whatsapl – Descrição

Esta aplicação desenvolvida em C++, tem como objectivo a gestão de uma aplicação messaging,como por exemplo Whatsapp e Facebook Messenger.

Esta aplicação permite gerir conversas entre utilizadores ou entre grupos de utilizadores.

Como solução para este desafio, criamos classes que nos facilitaram a resolução do mesmo.

1. Classe Date

Date representa um objecto referente a uma data,como por exemplo 2015/3/3.
Esta classe é bastante útil neste projecto visto que possibilita a datação de criação de grupos ou da entrada de um utilizador a um grupo,etc..

Possui os seguintes membros-dado:

- int *year*(ano)
- int *month*(mês)
- int *day*(dia)

Construtores:

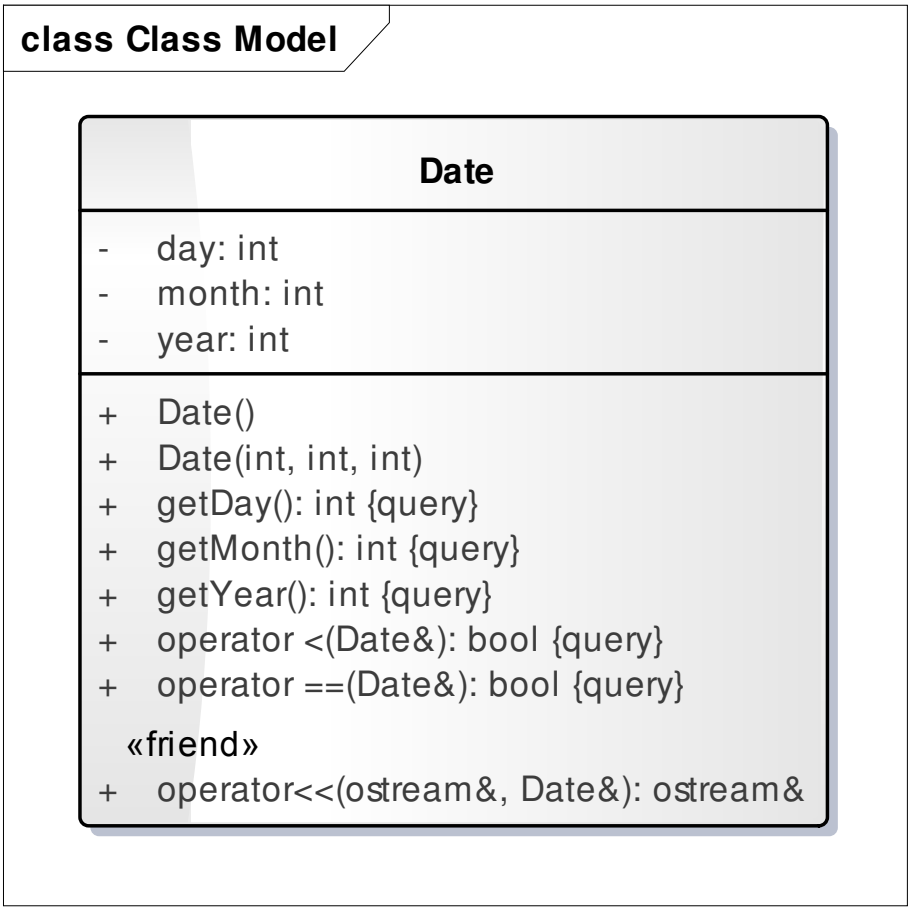
- *Date*() {}
- *Date*(int year, int month, int day)

Membros-função do tipo "get":

- int *getYear*() const;
- int *getMonth*() const;
- int *getDay*() const;

Possui ainda dois operadores == e <<, que permitem verificar se duas datas são iguais e imprimi-la, respectivamente:

- bool *operator* ==(Date d) const;
- friend ostream& *operator*<<(ostream &out, const Date &d);



2. Classe Time

Esta classe é uma classe derivada da classe Date.

Este objecto permite respresentar um minuto exacto no tempo, útil para saber o tempo no qual o utilizador enviou um mensagem.

Membros-dado:

- int *hour(hora)*
- int *minute*(minuto)

O contrutor desta função,usa o contrutor da função Date(herança):

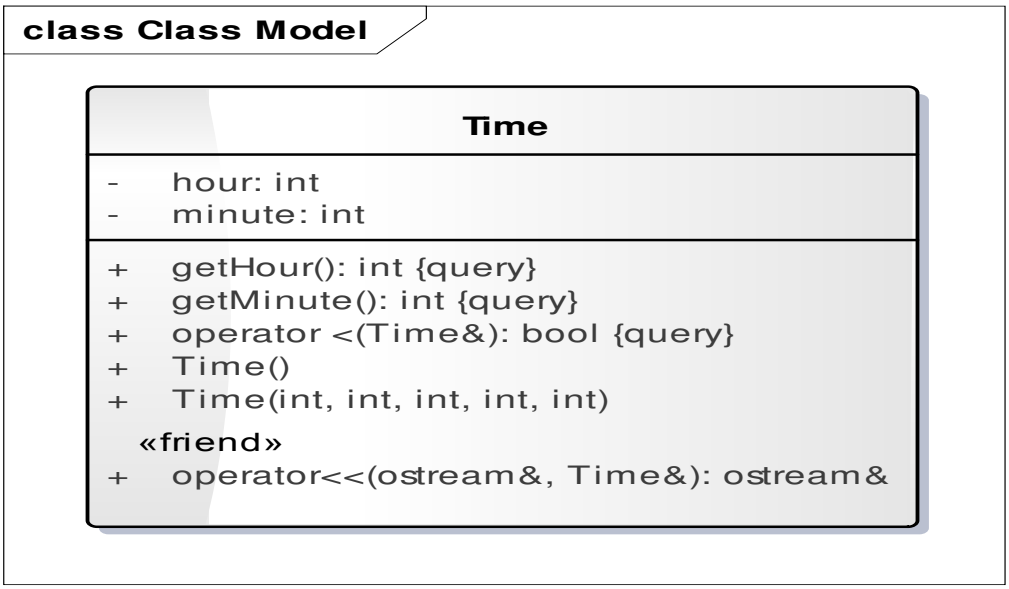
- *Time*() {};
- *Time*(int year, int month, int day, int hour, int minute);

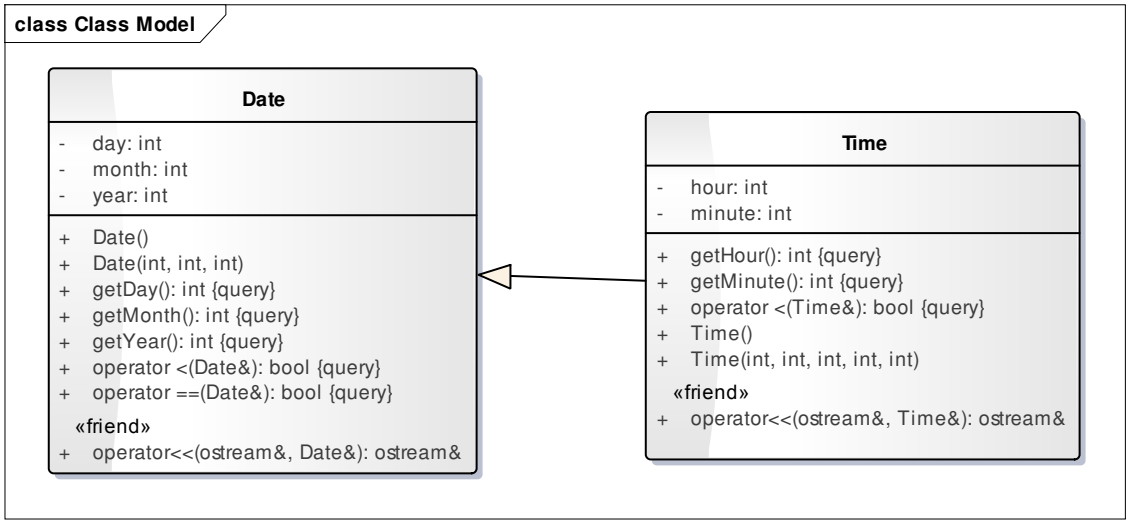
Membros-função "get":

- int *getHour*() const;
- int *getMinute*() const;

Operadores << e <, que representam respectivamente, uma função que imprime um exato minuto e uma função que permite verificar se um exato minuto é "menor" que outro:

- friend ostream& *operator*<<(ostream &out, const Time &t);
- friend bool *operator* < (Time &t1, Time &t2);





3. Classe User

Esta classe representa um utilizador da aplicação, sendo que cada objecto(utilizador) possui os seguintes membros-dado:

- string *username*(nome de utilizador)
- string *password*(palavra passe)
- string *name*(nome)
- string *email*(endereço eletrónico)
- long *cellphone*(número detelemóvel)
- date *join_date*(data de registo na aplicação)

Contrutores:

- *User*() {};
- *User*(string username, string password, string name, string email, long cellphone, Date join_date);

A Classe possui ainda membros-função do tipo "get", que permitem aceder aos mebros-dado private:

- string *getUsername()* const;
- string *getPassword()* const;
- string *getName()* const;
- string *getEmail()* const;
- long *getCellphone()* const;
- date *getJoinDate()* const;

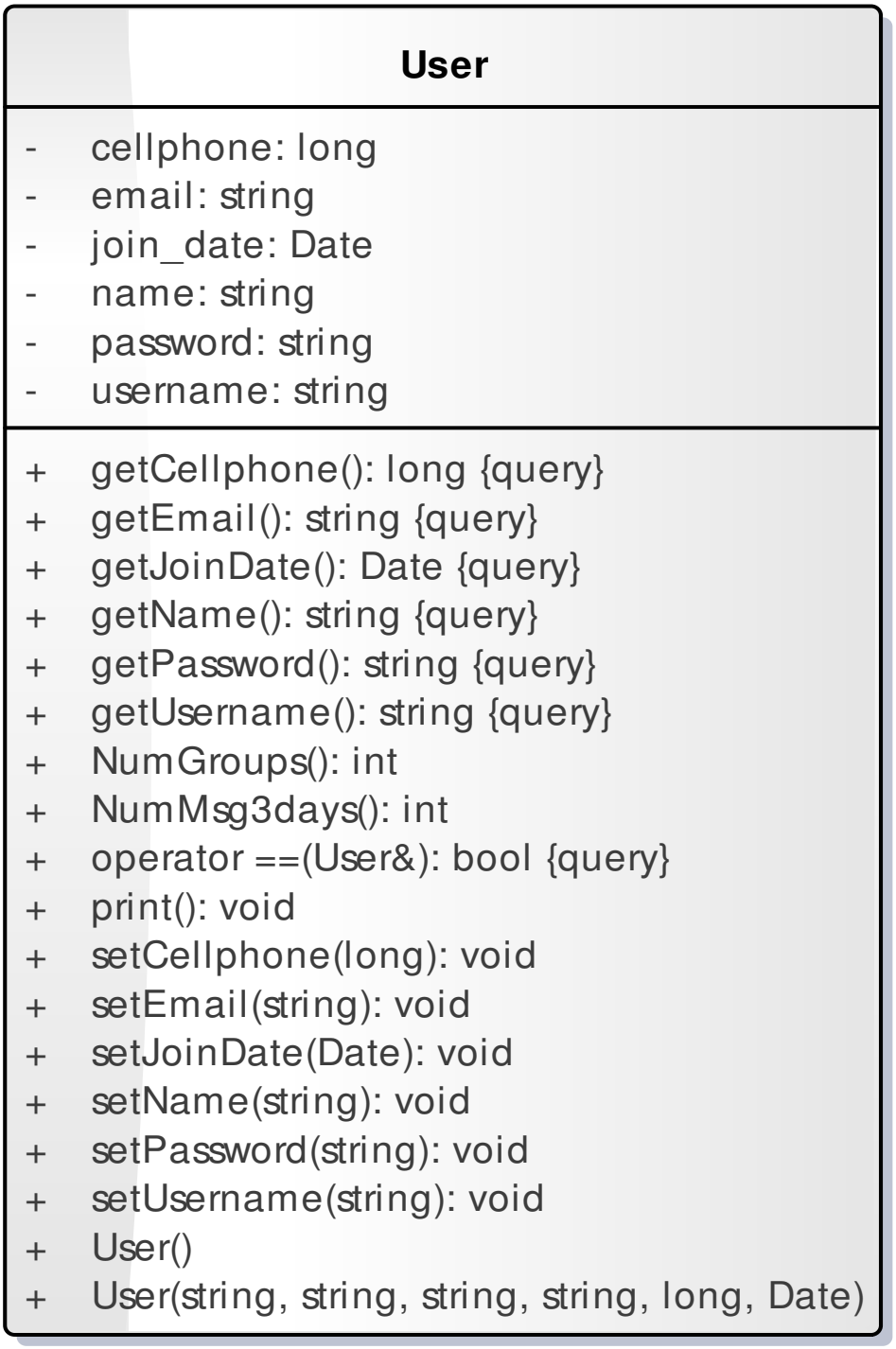
Membros-função do tipo "set" que permitem mudar os membros dado do objecto user,ou seja permitem mudar atributos de um utilizador, como o seu username,password,etc...

- void *setUsername*(string username);
- void *setPassword*(string password);
- void *setName*(string name);
- void *setEmail*(string email);
- void *setCellphone*(long cellphone);
- void *setJoinDate*(Date join_date);

Está ainda disponível a função que permitem imprimir toda informação de um utilizador, um operador que verifica se dois users são iguais e duas funções que retornam o número de grupos em que o utilizador está e o número de mensagens enviadas no últimos três dias.

- void *print*();
- bool *operator* ==(User &u) const;
- int NumGroups();
- int NumMsg3days();

class Class Model



4. Classe GroupUser

O objecto criado pela classe GroupUser representa um utilizador de um grupo.

A diferença para o objecto da classe User é que este possui a data de entrada num grupo tanto como o número de "bans"(expulsões).

Membros-dado(public):

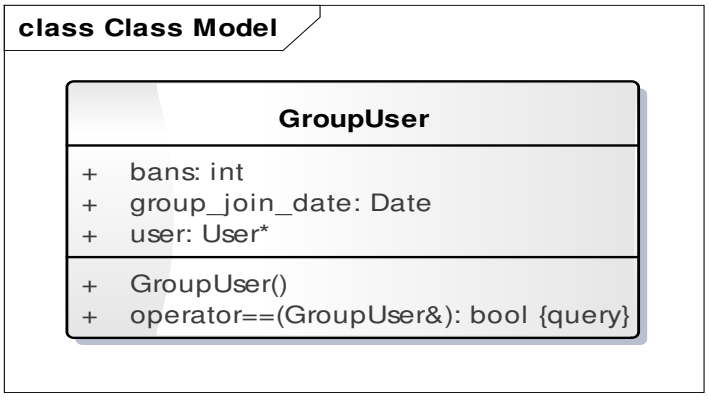
- User **user*;
- Date *group_join_date*;
- int *bans*;

O contrutor desta funcão apenas cria o objecto sem atribuir nenhum valor aos membros-dado, visto que estes podem ser acedidos naturalmente sem ter que recorrer a funções de "get":

- *GroupUser*() {};

Possui ainda um operador ==, que compara dois GroupUsers e verfica se são iguais:

- bool *operator*==(GroupUser &u) const;



5. Classe Message

O objecto desta classe encena uma mensagem enviada por um utilizador para outro utilizador ou para um grupo(conjunto de utilizadores).

Membros-dado:

- string *type(tipo de mensagem text/mms/mp4/etc)*
- string *content(mensagem)*
- Time *time_sent(minuto exacto da mensagem enviada)*
- User **sender(apontador para o remetente da mensagem)*
- GroupUser * group_sender;

Contrutores:

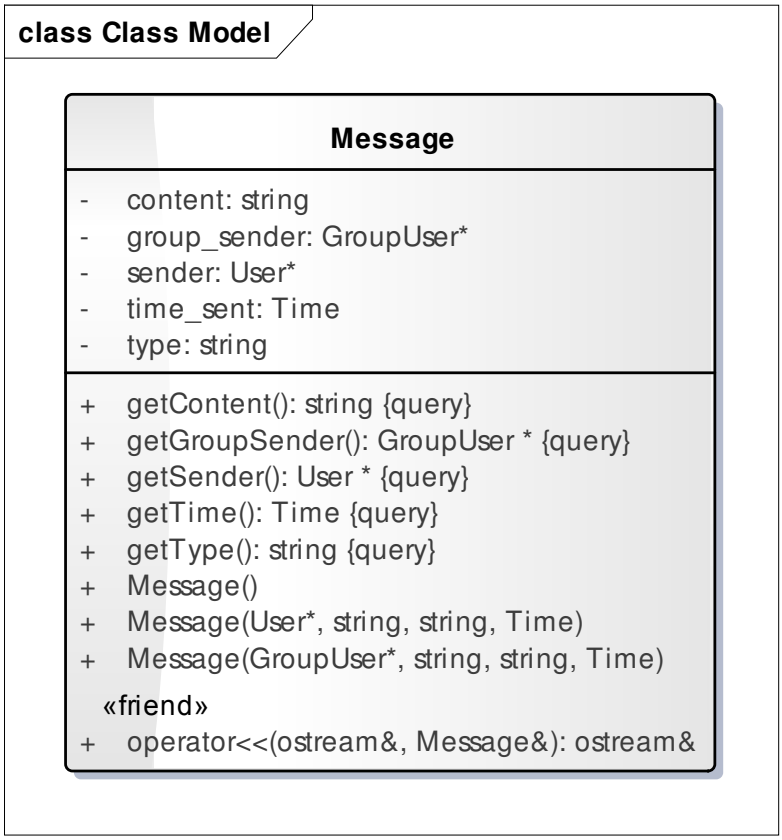
- *Message*() {};
- *Message*(User *sender, string type, string content, Time time_sent);
- *Message*(GroupUser *group_sender, string type, string content, Time time_sent);

Funções "get":

- string *getType()* const;
- string *getContent()* const;
- Time *getTime*() const;
- User **getSender*() const;

Operador:

- ostream& operator<<(ostream &out, const Message &m);



6. Classe Group

O objecto criado na classe Group é um grupo de conversação de n utilizadores.

Membros-dado:

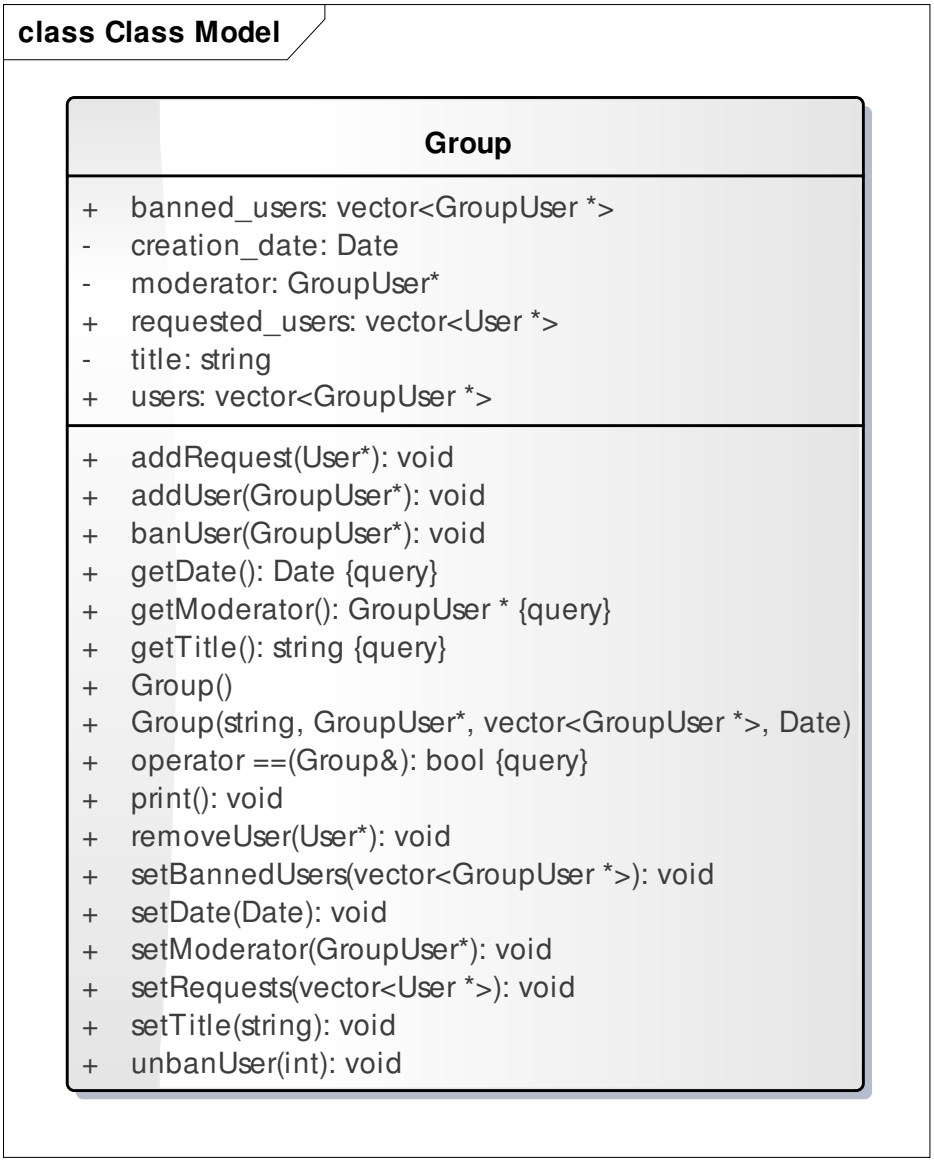
- string *title* (titlo do grupo)
- GroupUser **moderator*(apontador para o moderador do grupo)
- Date *creation_date*(data de criação do grupo)
- vector<GroupUser *> *users* (membros do grupo)
- vector<GroupUser *> *banned_users*(utilizadores banidos)
- vector<User *> *requested_users*(utilizadores que aguardam aceitação do moderador)

Membros-função “get” e “set”:

- string *getTitle()*const;
- GroupUser **getModerator*() const;
- Date *getDate*() const;
- void *setTitle*(string title);
- void *setModerator*(GroupUser *moderator);
- void *setDate*(Date creation_date);
- void *setBannedUsers*(vector<GroupUser *> banned_users);
- void *setRequests*(vector<User *> requested_users);

Possui ainda funções que adicionam/removem /ban/unban utilizadores, adicionam utilizadores á lista de utilizadores pendentes, imprimem a informação referente a ao grupo e um comparam grupos(operador):

- void *addUser*(GroupUser *user);
- void *removeUser*(User *u);
- void *banUser*(GroupUser *u);
- void *unbanUser*(int pos);
- void *addRequest*(User *u);
- void *print*();
- bool *operator* ==*(*Group &g) const;



7. Classe Conversation

O objecto referente a classe Conversation , caracteriza uma conversa entre dois utilizadores ou entre vários utilizadores num grupo.

Membros-dado:

- vector<Message *> *messages*(conjunto de mensagens trocadas pelos utilizadores)
- Group * *group*(caso seja apenas entre dois utlizadores o apontador é NULL)

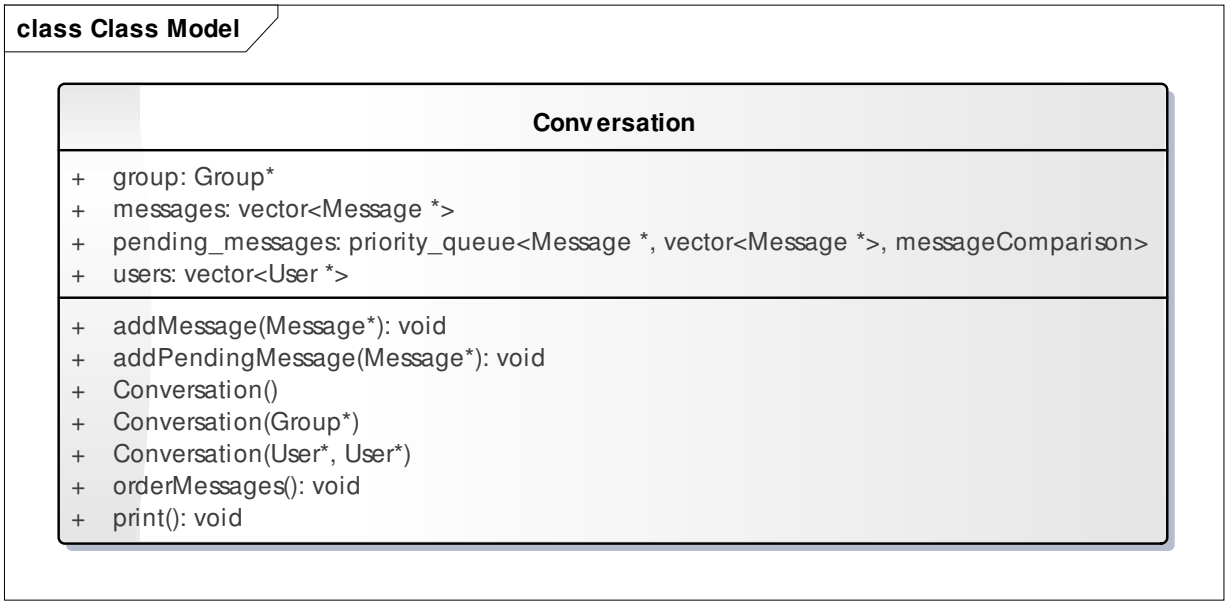
- vector<User *> *users*(utilizadores da conversação)
- priority_queue<Message *, vector<Message *>, messageComparison> pending_messages (mensagens por aprovar pelo utilizador)

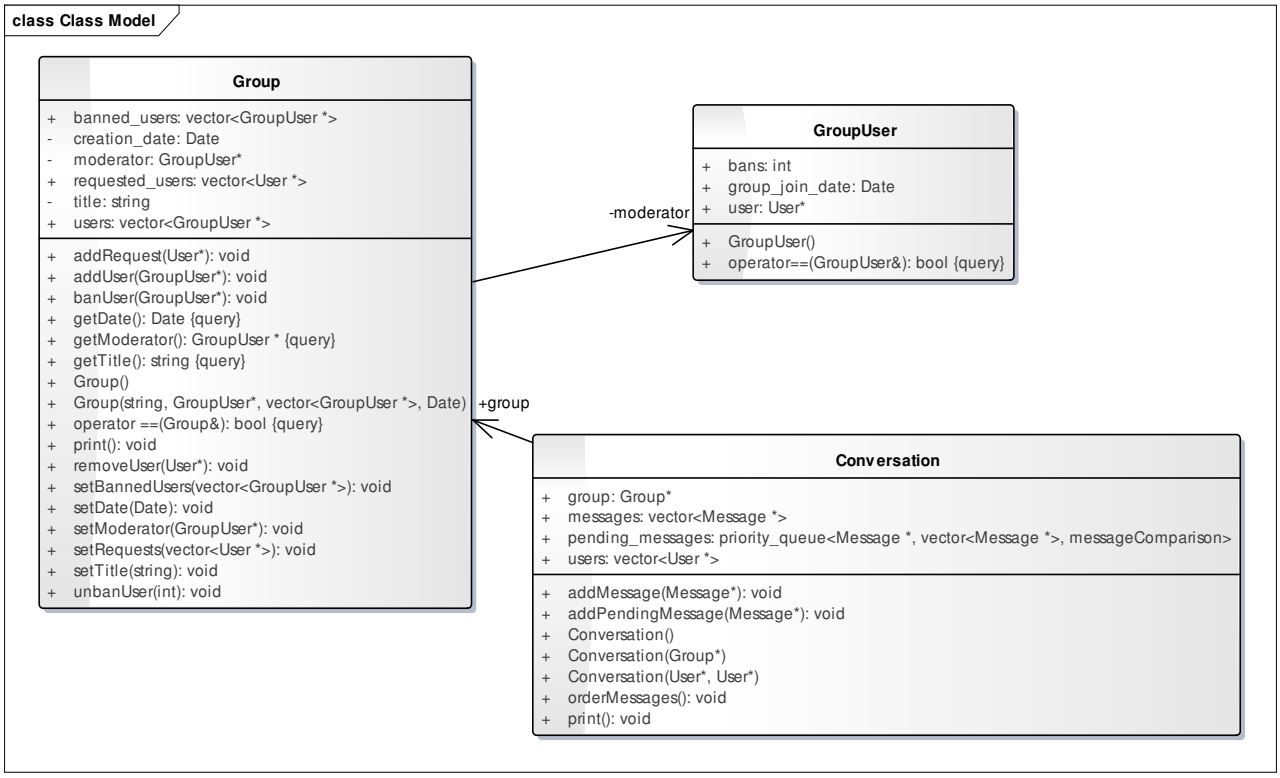
Contrutores:

- *Conversation*() {};
- *Conversation*(Group *group); (caso seja uma conversação num grupo)
- *Conversation*(User *user1, User *user2); (caso seja uma conversação entre dois utilizadores)

Possui dois membros-função importantes.Uma ordena todas as mensagens por ordem cronologica,ou seja, mesmo quando as mensagens lidas no ficheiro estejam desordenadas a função ordena-as, outra imprime uma conversa:

- void *orderMessages*();
- void *print*();





8. Classe Database

A classe Database funciona como a base dados enquanto a aplicação corre. Possui todos os utilizadores, grupos e conversações:

- vector<User *> *users*
- vector<Group *> *groups*
- vector<Conversation *> *conversations*
- BST<UserPtr> favusers
- tr1::unordered_set<User *, hstr, eqstr> inactive_users

Contrutor:

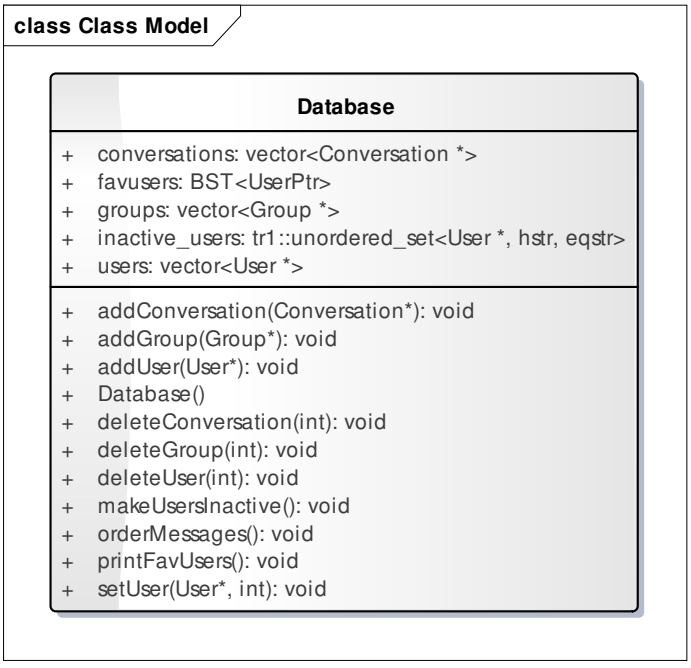
- *Database*();

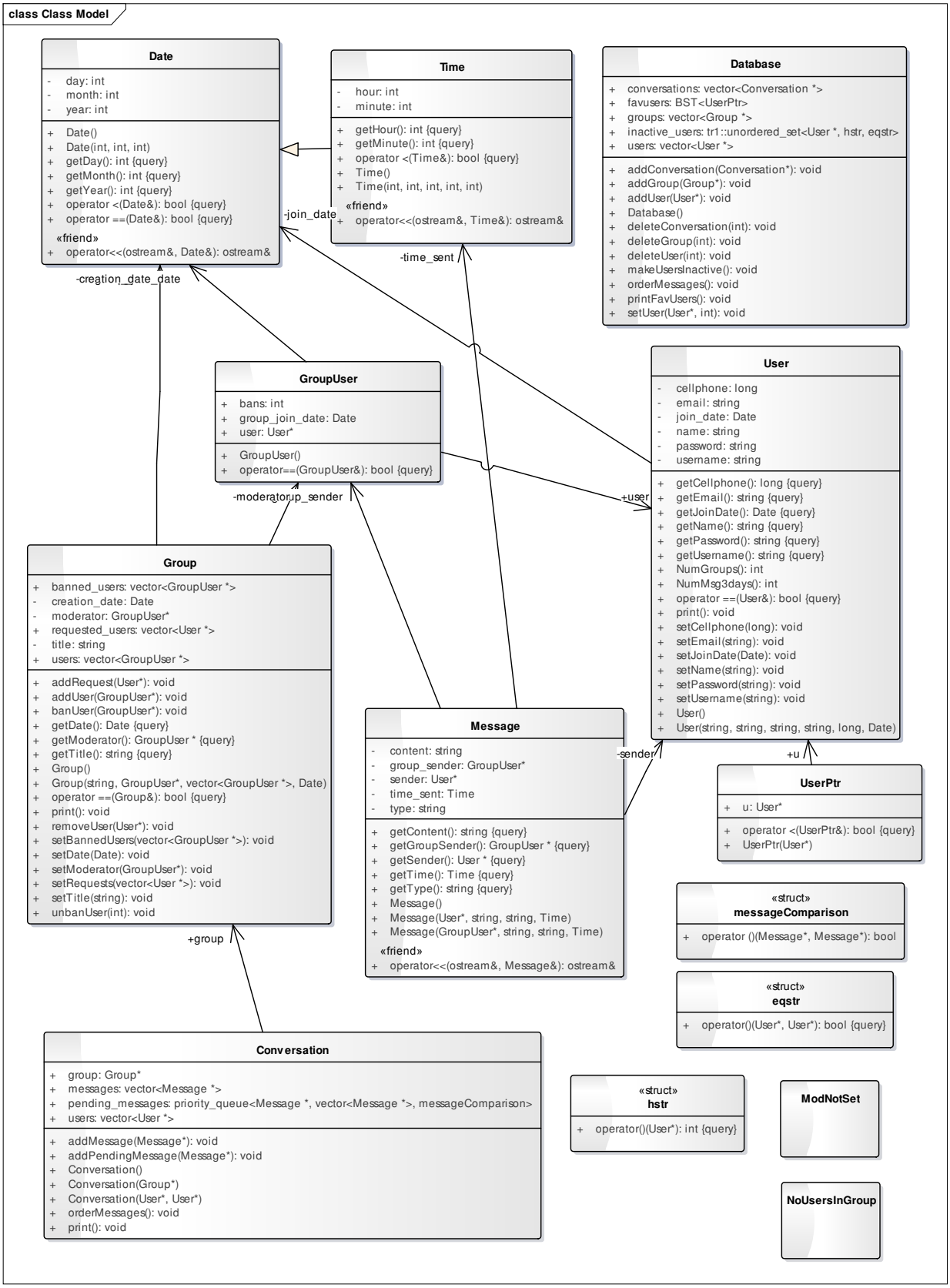
Os membros função permitem adicionar e apagar utilizadores, grupos e conversações e tammbém mudar um utilizador numa determinada posição:

- void *addUser*(User *u);
- void *addGroup*(Group *g);
- void *addConversation*(Conversation *c);
- void *deleteUser*(int pos);
- void *deleteGroup*(int pos);
- void *deleteConversation*(int pos);
- void *setUser*(User *u, int pos);

Outras funções:

- void orderMessages(); (Ordena todas as conversações)
- void printFavUsers();(Imprime os utilizadores favoritos)
- void makeUsersInactive()(Coloca todos os utilizadores que não utilizam a aplicação á mais de 30 dias)




class Class Model
Date
- day: int
- month: int
- year: int
+ Date()
+ Date(int, int, int)
+ getDay(): int {query}
+ getMonth(): int {query}
+ getYear(): int {query}
+ operator <(Date&): bool {query}
+ operator ==(Date&): bool {query}
«friend»
+ operator<<(ostream&, Date&): ostream&
Time
- hour: int
- minute: int
+ getHour(): int {query}
+ getMinute(): int {query}
+ operator <(Time&): bool {query}
+ Time()
+ Time(int, int, int, int, int)
«friend»
+ operator<<(ostream&, Time&): ostream&
Database
+ conversations: vector<Conversation *>
+ favusers: BST<UserPtr>
+ groups: vector<Group *>
+ inactive_users: tr1::unordered_set<User *, hstr, eqstr>
+ users: vector<User *>
+ addConversation(Conversation*): void
+ addGroup(Group*): void
+ addUser(User*): void
+ Database()
+ deleteConversation(int): void
+ deleteGroup(int): void
+ deleteUser(int): void
+ makeUsersInactive(): void
+ orderMessages(): void
+ printFavUsers(): void
+ setUser(User*, int): void
-join_date
-time_sent
-creation_date_date
GroupUser
+ bans: int
+ group_join_date: Date
+ user: User*
+ GroupUser()
+ operator==(GroupUser&): bool {query}
-moderatorup_sender
+user
User
- cellphone: long
- email: string
- join_date: Date
- name: string
- password: string
- username: string
+ getCellphone(): long {query}
+ getEmail(): string {query}
+ getJoinDate(): Date {query}
+ getName(): string {query}
+ getPassword(): string {query}
+ getUsername(): string {query}
+ NumGroups(): int
+ NumMsg3days(): int
+ operator ==(User&): bool {query}
+ print(): void
+ setCellphone(long): void
+ setEmail(string): void
+ setJoinDate(Date): void
+ setName(string): void
+ setPassword(string): void
+ setUsername(string): void
+ User()
+ User(string, string, string, string, long, Date)
Group
+ banned_users: vector<GroupUser *>
- creation_date: Date
- moderator: GroupUser*
+ requested_users: vector<User *>
- title: string
+ users: vector<GroupUser *>
+ addRequest(User*): void
+ addUser(GroupUser*): void
+ banUser(GroupUser*): void
+ getDate(): Date {query}
+ getModerator(): GroupUser * {query}
+ getTitle(): string {query}
+ Group()
+ Group(string, GroupUser*, vector<GroupUser *>, Date)
+ operator ==(Group&): bool {query}
+ print(): void
+ removeUser(User*): void
+ setBannedUsers(vector<GroupUser *>): void
+ setDate(Date): void
+ setModerator(GroupUser*): void
+ setRequests(vector<User *>): void
+ setTitle(string): void
+ unbanUser(int): void
Message
- content: string
- group_sender: GroupUser*
- sender: User*
- time_sent: Time
- type: string
+ getContent(): string {query}
+ getGroupSender(): GroupUser * {query}
+ getSender(): User * {query}
+ getTime(): Time {query}
+ getType(): string {query}
+ Message()
+ Message(User*, string, string, Time)
+ Message(GroupUser*, string, string, Time)
«friend»
+ operator<<(ostream&, Message&): ostream&
-sender
+u
UserPtr
+ u: User*
+ operator <(UserPtr&): bool {query}
+ UserPtr(User*)
«struct»
messageComparison
+ operator ()(Message*, Message*): bool
«struct»
eqstr
+ operator()(User*, User*): bool {query}
+group
Conversation
+ group: Group*
+ messages: vector<Message *>
+ pending_messages: priority_queue<Message *, vector<Message *>, messageComparison>
+ users: vector<User *>
+ addMessage(Message*): void
+ addPendingMessage(Message*): void
+ Conversation()
+ Conversation(Group*)
+ Conversation(User*, User*)
+ orderMessages(): void
+ print(): void
«struct»
hstr
+ operator()(User*): int {query}
ModNotSet
NoUsersInGroup

Whatsapl – Casos de utilização

1. Main Menu

O menu principal tem a seguinte apresentação:

```
==================================================
WHATSAPL

1 - Users
2 - Groups
3 - Conversations
4 - Save

0 - Save and Exit
```

Opções:

1.Users (Menu dos Utilizadores)

2.Groups (Menu dos Grupos)

3. Conversations (Menu das conversas)

4. Save(Guarda todos as alterações feitas até ao momento nos ficheiros de texto)

0. Save and Exit (Guarda todo as mudanças efectuadas para os ficheiros e termina o programa)

2. Users Menu

O menu dos utilizadores é apresentado da seguinte maneira:

```
==================================================
USERS

1 - Create user
2 - View users
3 - Edit user
4 - Delete user

0 - Back
```

2.1 **Create User** (Menu de criação de um utilizador)

```
================================================
CREATE USER
Username: Exemplo
Password: relatorio
Confirm password: relatorio
Name: AEDA
Email: aeda@gmail.com
Cellphone: 911111111
Join date
   Year: 2015
   Month: 11
   Day: 8

User created
```

2.2 **View Users** (permite ver os utilizadores já criados)

```
================================================
VIEW USERS

1 - Active users
2 - Most active users
3 - Inactive users

0 - Back
```

```
================================================
VIEW USERS

1 - Active users
2 - Most active users
3 - Inactive users

0 - Back
1
Username: Exemplo
Password: relatorio
Name: AEDA
Email: aeda@gmail.com
Cellphone: 911111111
Join date: 2015/12/23

0 - Back
```

```
==================================================
VIEW USERS

1 - Active users
2 - Most active users
3 - Inactive users

0 - Back
2

1 - Exemplo - 0 messages

0 - Back
```

2.3 Edit Users (Permite mudar os utilizadores)

```
==================================================
SELECT USER TO EDIT

1 - Search username

0 - Back
1


Username: antonio


==================================================
EDIT USER

1 - Username
2 - Password
3 - Name
4 - Email
5 - Cellphone
6 - Date joined
7 - Make active

0 - Back
```

Neste exemplo podemos alterar todos as componentes do utilizador antonio como username/password/name/torná-lo activo/etc...

2.4 **Delete User** (Permite eliminar um utilizador)

```
==================================================
SELECT USER TO DELETE

1 - Search username

0 - Back
1


Username: antonio

Username: antonio
Password: 1234
Name: antonio
Email: antonio@gmail.com
Cellphone: 5678
Join date: 2015/11/8

Delete user?
1 - Yes
2 - No
1

User deleted
```

2.5 Back

Voltamos ao menu principal.

3. Groups Menu

```
==================================================
GROUPS

1 - Create group
2 - View groups
3 - Edit group
4 - Delete groups

0 - Back
```

3.1 **Create group** (Cria um grupo)

```
==================================================
CREATE GROUP

Title: AEDA


Group creation date

   Year: 2015
   Month: 3
   Day: 3


==================================================
SELECT GROUP MODERATOR

1 - Search username

0 - Back
1


Username: bruno
Group join date
   Year: 2015
   Month: 3
   Day: 3
```

```
==================================================
ADD GROUP USERS

1 - Search username

0 - Back
1


Username: tofran
Group join date
   Year: 2015
   Month: 3
   Day: 3


==================================================
ADD GROUP USERS

1 - Search username

0 - Back
0

Group created
```

3.2 **View Groups** (permite visualizar todos os grupos)

```
==================================================
VIEW GROUPS

Title: League of Legends
Moderator: bruno
Users: [bruno, 2015/11/8, 0],  [vasco, 2015/11/8, 0],  [luis, 2015/11/8, 0]
Banned users: []
Requests: [tofran]
Creation Date: 2015/11/8

Title: CS:GO
Moderator: tofran
Users: [tofran, 2015/11/8, 0],  [vasco, 2015/11/8, 0]
Banned users: [luis]
Requests: [bruno]
Creation Date: 2015/11/8

Title: FEUP
Moderator: bruno
Users: [bruno, 2015/11/8, 0],  [luis, 2015/11/8, 0]
Banned users: []
Requests: []
Creation Date: 2015/11/8

Title: AEDA
Moderator: bruno
Users: [bruno, 2015/3/3, 0],  [tofran, 2015/3/3, 0]
Banned users: []
Requests: []
Creation Date: 2015/3/3

0 - Back
```

3.3 **Edit Groups** (permite editar um grupo)

```
================================================
EDIT GROUPS

1 - Search title

0 - Back
1


Title: FEUP


================================================
EDIT GROUPS

1 - Title
2 - Moderator
3 - Users
4 - Creation date

0 - Back
```

3.4 **Back**

Voltamos ao menu principal.

4. Conversations Menu

```
==================================================
CONVERSATIONS

1 - Create conversation
2 - View conversations
3 - Edit conversation
4 - Delete conversation

0 - Back
```

4.1 **Create conversation** (permite criar uma conversa entre utilizadores)

```
==================================================
CREATE CONVERSATION

1 - Group
2 - Private

0 - Back
```

4.1.1 Group

```
==================================================
CREATE CONVERSATION

1 - Group
2 - Private

0 - Back
1


Title: AEDA
```

4.1.2 Users

```
==================================================
CREATE CONVERSATION

1 - Group
2 - Private

0 - Back
2


User 1
Username: bruno

User 2
Username: tofran
```

4.2 **View Conversations** (Permite ver todas as conversas)

```
==================================================
VIEW CONVERSATIONS

1 - Group: [CS:GO]
2 - Group: [FEUP]
3 - Group: [League of Legends]
4 - Private: [antonio, bruno]
5 - Private: [bruno, tofran]

0 - Back
1


==================================================
VIEW CONVERSATION

tofran - 2015/11/8, 1:52
text: sou dmg!!!

vasco - 2015/11/8, 1:53
text: nice

antonio - 2015/11/8, 1:53
text: pff not even global


0 - Back
```

4.3 **Edit Conversations** (permite adicionar mensagens a uma conversa ou o moderador aprovar as mensagens enviadas para o grupo)

```
==================================================
EDIT CONVERSATION

1 - Group: [CS:GO]
2 - Group: [FEUP]
3 - Group: [League of Legends]
4 - Private: [antonio, bruno]
5 - Private: [bruno, tofran]

0 - Back
1


==================================================
EDIT CONVERSATION

1 - Add message
2 - Approve messages

0 - Back
1
  ==================================================
  EDIT CONVERSATION

  1 - Add message
  2 - Approve messages

  0 - Back
  1


  ==================================================
  ADD MESSAGE TO CONVERSARTION

  Sender username: tofran
  Type: txt
  Content: global rsrsrsrs
  Time sent
     Year: 2015
     Month: 12
     Day: 23
     Hour: 19
     Minute: 18

  Add message?
  1 - Yes
  2 - No
  1
```

4.4 Delete Conversations (permite apagar um conversa)

4.4.1 Group

```
==================================================
DELETE CONVERSATION

1 - Group
2 - Private

0 - Back
1


Title: FEUP

Group: [FEUP]

Delete conversation?
1 - Yes
2 - No
1

Conversation deleted
```

4.4.2 Users

```
==================================================
DELETE CONVERSATION

1 - Group
2 - Private

0 - Back
2


User 1
Username: bruno

User 2
Username: tofran

Private: [bruno, tofran]

Delete conversation?
1 - Yes
2 - No
1

Conversation deleted
```

4.4 Back

Voltamos ao menu principal.

Whatsapl – Conclusão

Visto que esta aplicação deve apenas gerir e não implementar um aplicação messaging, as funções que permitem o moderador aceitar/rejeitar os pedidos de adesão de utilizadores, bloquear/desbloquear membros e registar os bloqueamentos, não estão disponíveis na "user interface" apesar de estarem implementadas no código e operacionais no caso da criação da aplicação.

Para podermos saber quais os utilizadores mais "fevorosos", sempre que for chamada a função para os imprimir, coloca todos os utilizadores num **árvore binária de pesquisa**. Colocando os utilizadores com mais mensagens nos últimos três dias em primeiro lugar.

Usamos também uma **fila de prioridade** para guardar as mensagens que ficam por validar pelo moderador.

A **tabela de dispersão** é usada para guardar utilizadores inactivos, ou seja, com mais de 30 dias de inactividade.

As principais díficuldades encontradas no desenvolvimento deste projecto foram:

- Criação de funções auxiliares para evitar a repetição de código
- Gestão e verificação de input para evitar comportamentos desejados
- Alteração de código para lidar com problemas que surgiram que não se tinha planeado previamente

Pensamos que este trabalho foi bastante enriquecedor, visto que nos fez perceber os pontos onde tinhamos mais dificuldades ultapassando-os com sucesso.

O esforço colocado neste trabalho foi total. Todos os membro contribuíram igualmente para o desenvolvimento

deste projecto.